Performance Monitoring and Management A ‘SaaS in a Box’ Tutorial

# Purpose of this tutorial

The Wingtip SaaS app is built using single-tenant data model, with each venue (tenant) having their own database. Like many SaaS applications, the anticipated tenant workload pattern is unpredictable and sporadic – ticket sales may occur at any time, and event administration is only an occasional activity. Consequently, the tenant databases are deployed by default into elastic database pools, which optimizes the cost of the solution by sharing resources across many databases. With this kind of pattern, it’s important to monitor database and pool resource usage to ensure that loads are reasonably balanced among pools, that individual databases have adequate resources, and that pools are not running hot. This tutorial explores ways to monitor and manage databases and pools, and shows how to take corrective action in response to variations in workload.

# Introduction to the SaaS Performance Management Patterns

Elastic database pools provide a cost-effective way to provide resources for a group of databases that have unpredictable workloads. With the right workload pattern, even two databases can benefit from being managed in a pool. Not only does a pool share the cost of resources, it can also remove the need to monitor individual databases and scale them individually to track variations in workload.



Pools, and the elastic database they contain, still need to be monitored to ensure they stay within acceptable ranges of performance. Pool configuration can be tuned to meet the needs of the aggregate workload, ensuring that the pool DTUs are appropriate for the overall workload, and that the per-database min and max DTU values are appropriately set given the application requirements.

To avoid being glued to a monitoring dashboard, it’s most effective to **set alerts that fire if databases or pools stray out of normal ranges**.

To respond to short term fluctuations in the aggregate performance level of a pool the **pool eDTU level can be scaled up or down**.

Where this fluctuation occurs on a regular predictable basis, **scaling of the pool can be scheduled to occur automatically** – overnight or during weekends.

To respond to longer term fluctuations, or changes in the database population, **databases can be moved into other pools**.

To respond to short term increases in *individual* database load (which may be caused by tenant activity or database maintenance tasks), **individual databases can be taken out of a pool and assigned an individual performance level** for a period. Once the load is reduced the database can then be returned to the pool. Where this is known in advance databases can be moved pre-emptively to ensure the database always has the resources it needs and avoid impact on other databases in the pool. If this requirement is predictable – such as a venue experiencing a rush of ticket sales for a popular event – then this management behavior can be integrated in the application – enabling **tenant self-service performance adjustments** – and even linked into the tenant billing model.

# Monitoring and managing performance

The Azure portal provides built-in monitoring and alerting on most resource blades. For SQL Database, monitoring and alerting is available on databases and pools. This built-in monitoring and alerting is resource-specific. It’s convenient to use for small numbers of resources, but is less so for large numbers. And there is a limit of 1000 resources having alerts per subscription.

For high-volume scenarios Log Analytics (also known as OMS) can be used. This is a separate Azure service that provides analytics over emitted diagnostic logs and telemetry gathered in a log analytics workspace, which can collect telemetry from many services and be used to query and set alerts.

The tutorial walks through the built-in monitoring and alerting features and then explores several of the key performance management scenarios described above. A separate tutorial covers setting up and using Log Analytics.

# Setup

Download and extract **WTPLearningModules.zip** to a convenient folder.

Deploy the **WTP Application.** Ensurethe catalog is initialized using the Demo Assistant app. See the Introduction to the WTP SaaS Application tutorial for deployment instructions.

**SSMS** can be used to explore database schema and execute SQL queries directly.

**PowerShell ISE** is recommended to execute scripts and follow their execution in debug mode.

**PowerShell Tips**

* Open and configure demo- scripts in the PowerShell ISE.
* Use F5 to run the script (using F8 is not advised as the $PSScriptRoot variable is not evaluated when running snippets of a script).
* Use F9 to set a breakpoint to let you trace the script in debug mode to see how it works
* Use F10 to step through the script, F11 to step into a function, and Shift-F11 to step out.

To save time, it’s recommended to **pre-install the batch of tenants** (this is described at the end of the Provision and Catalog tutorial, as well as in this tutorial below).

# Walkthrough

## Getting started – provision a batch of additional tenants.

This tutorial requires you have more databases deployed to get an understanding of how performance monitoring and management works at scale. While pools can be cost-effective with just two database, the more databases that are in the pool the pronounced the averaging effect becomes. A minimum of 20 tenant databases is suggested, but even then, aggregate pool loads will still tend to be very spikey.

1. Update the user configuration file used by all tutorial scripts. Update again if you redeploy the app.
   1. Open ...\Learning Modules\UserConfig.psm1 in **PowerShell ISE**
   2. Modify **$userConfig.ResourceGroupName** to the resource group used for the deployed app.
   3. Modify **$userConfig.Name** to the User name used for the deployed app.
2. Open …\Learning Modules\Provision and Catalog\**Demo-PerformanceMonitoringAndManagement**.ps1 in PowerShell ISE.Keep this script open as you’ll run several of the scenarios it supports during this tutorial.
3. Modify **$DemoScenario** to **1, Provision a batch of tenants**
4. Execute the script using **F5**.
5. The script will deploy a batch of 17 additional tenants. It takes a few minutes to run, so it’s best done prior to the tutorial.

The New-TenantBatch script uses a nested or linked set of ARM templates that create a batch of tenants, which by default copies the database **baseTenantDb** on the catalog server to create the new tenant databases, then registers these in the catalog, and finally initializes them with the tenant name and venue type. This is consistent with the way the WTP app provisions a new tenant. Any changes made to baseTenantDB will be applied to any new tenants provisioned thereafter. See the Schema Management tutorial to see how to make schema changes to the existing tenant databases.

## Getting started – start the load generator

Once the new databases are fully provisioned, start the load generator to apply a load to all of your tenant databases.

1. Set $DemoScenario = 2, **Generate a normal intensity load**
2. Execute the script to star the load generation jobs using F5.
3. Given the sporadic nature of the load, it takes a few minutes for the activity to achieve a kind of steady state.

The load generator applies a ‘synthetic’ CPU-only load to every database on the tenant. While it is not like the load that would be produced by the application, it is adequate for this tutorial. The generator starts a job for each tenant database, which calls a stored procedure periodically, which generates the load. The load levels (in DTUs), duration, and intervals are varied across all databases, simulating unpredictable tenant activity.

IMPORTANT. The load generator is running as a series of jobs in your local PowerShell session. If you close PowerShell, or the tab it was started in, or suspend your machine the jobs will stop.

## Exercise 1: Monitor resource usage on pools and databases in the portal

To monitor the resource usage that results from the load being applied, open the portal on the pool containing the tenant databases.

1. Open the Azure portal and locate the customers1-<USER> server in the WTP resource group.
2. You should see the list of tenant databases including the newly created databases.
3. Scroll down the blade and locate the elastic pools part and click on Pool1. This pool contains all the tenant databases created so far.
4. Expand the pool blade that opens and observe the pool utilization chart and top database utilization chart.

The pool utilization is essentially the aggregate of the database utilization for all databases in the pool.

The database utilization chart shows the hottest 5 databases. As there are other databases in the pool beyond the top 5, the pool utilization will show activity that is not reflected in the top 5 databases chart.

## Exercise 2: Set alerts on the pool

Set an alert on the pool that will trigger on > 75% utilization sustained for 5 minutes as follows:

1. Open the pool blade for the Customers1/Pool1 in the Azure portal.
2. Select **Alert Rules** in the context menu.
3. On the alert rules list, click on the **+ Add alert** command.
4. Provide a name, such as ‘**High DTU**’, and a description if desired.
5. Set **Metric = eDTU percentage**.
6. Set **Condition = greater than**.
7. Set **Threshold = 75**.
8. Set **Period = Over the last 30 minutes**.
9. Note that you can have notifications sent to your Azure account email, and optionally other additional emails (recommend not to set this unless you own the subscription being used).

Note: As the alert will only fire if the threshold is exceeded for the last 30 minutes, it’s unlikely that you see any alerts during this exercise regardless of the current levels of activity.

A web hook can also be specified to cause an action to be taken on the alert.

## Exercise 3: Scale up a busy pool

If the aggregate load level increases on a pool to the point that it reaches 100% DTU then individual database performance will be affected, potentially slowing query response times for all databases in the pool. Short term options include scaling up the pool to provide additional resources, removing databases from the pool – moving them to other pools or to a stand-alone service tier. Longer term, consider optimizing queries or index usage to improve database performance. Depending on the application’s sensitivity to performance issues it’s best practice to scale a pool up before it reaches 100%. Use an alert to warn you well in advance.

You can simulate a busy pool by increasing the load produced by the generator. Causing the databases to burst more frequently and for longer will increase the aggregate load on the pool without changing the requirements of the individual databases. Scaling up the pool is easily done in the portal or from PowerShell. This exercise uses the portal.

1. In the …\**Demo-PerformanceManagementAndMonitoring**.ps1 script
2. Set **$DemoScenario = 3, Generate load with longer and more frequent bursts per database** to increase the intensity of the aggregate load on the pool without changing the peak load required by each database.
3. Execute the script using **F5**.
4. **Open the pool blade** **for Customers1/Pool1**.
5. Monitor the increased pool DTU usage on the upper chart. It will take a few minutes for the new higher load to kick in, but you should quickly see that the pool starts to hit 100% utilization, and as the load steadies into the new pattern it will rapidly overload the pool.
6. To scale up the pool, click **Configure pool**
7. Adjust the **Pool eDTU** slider to 100 (recommend you go no higher to limit costs). Note how the aggregate storage available for all databases in the pool, indicated by **Pool GB**, is linked to the eDTU setting and increases also. Changing the pool DTU does not change the per-database settings (which will still be at 50 DTU max per-database DTU). You can see the per-database settings on the far right tab of the **Configure pool** blade.
8. Click the **Save** command to submit the request. The change will typically take 3-5 minutes for a Standard pool.
9. Monitor the effect of providing the pool with more resources (although with few databases and a randomized load it’s not always easy to see conclusively). While you are looking at the charts bear in mind that 100% on the upper chart now represents 100 eDTUs, while on the lower chart 100% is still 50 eDTUs as the per-database max is still 50 eDTUs.

Databases remain online and fully available throughout the process. At the last moment as each database is ready to be enabled with the new pool DTU, any active connections are broken. Application code should always be written to retry dropped connections, and so will reconnect to the database in the scaled-up pool.

## Exercise 5: Create a second pool and load-balance databases to handle increased aggregate load

As an alternative to scaling up the pool, create a second pool and move databases into it to balance the load between the two pools. To do this the new pool must be created on the same server as the first.

1. Open the **server blade for the customers1-<USER> server**. If you are on a database or pool blade, you can drop down the essentials control and select the server name as a shortcut.
2. Click **+ New pool** to create a pool on the current server
3. On the New Elastic database pool template:
   1. set **Name = Pool2**.
   2. Leave the pricing tier as **Standard Pool**.
   3. Click on **Configure pool**,
   4. On the Configure Pool blade that opens, set **Pool eDTU = 50 DTU**.
   5. Click the **Add databases** command to see a list of databases on this server that are not in the current pool.
   6. On the list, **check** half of the databases (10 out of 20) to move these to the new pool, and then click **Select**.
   7. Click **Select** again to accept the configuration changes. Note the cost estimate for one month of usage with the selected options.
   8. Select **OK** to create the new pool with the new configuration and to move the databases.

Creating the pool and moving the databases into it takes a few minutes. Each of the databases being moved remains online and fully accessible until the last moment, at which point any open connections are closed. When a client retries the connection, it will connect to the database in the new pool.

Once the pool has been created it will appear on the customers1 server blade. Click on the pool name to open the pool blade and monitor its performance.

You should see that resource usage on Pool1 has dropped and that Pool2 is similarly loaded.

## Exercise 6: Manage an increased load on a single database

If a single database in a pool experiences a sustained high load, depending on the pool configuration, it may tend to dominate the resources in the pool and impact other databases. If the activity is likely to continue for some time, the database can be moved out of the pool temporarily. This both allows the database to be given more resources than the others in the pool, and isolates it from the other databases. This exercise simulates the effect of Contoso Concert Hall experiencing a high load when tickets go on sale for a popular concert.

1. In the …\**Demo-PerformanceManagementAndMonitoring**.ps1 script
2. Set **$DemoScenario = 5, Generate a normal load plus a high load on a single tenant (approx 95 DTU).**
3. Set **$SingleTenantDatabaseName = contosoconcerthall**
4. Execute the script using **F5**.
5. **Open the pool blade** **for Customers1/Pool1**.
6. Look at the **Elastic pool monitoring** display at the top of the blade and look for the increased pool DTU usage. After a minute or two, the higher load should start to kick in, and you should quickly see that the pool hits 100% utilization.
7. Also monitor the **Elastic database monitoring** display which shows the hottest databases in the past hour. The contosoconcerthall database should soon appear as one of the 5 hottest databases.
8. **Click on the Elastic database monitoring** **chart** and it will open a **Database Resource Utilization** blade on which you can selectively monitor any of the databases. This lets you isolate the display for the contosoconcerthall database.
9. From the list of databases, **click on contosoconcerthall** and its database blade will open.
10. Click on **Pricing Tier (scale DTUs)** in the context menu to open the **Configure performance** blade on which you can set an isolated performance level for the database.
11. Click on the **Standard** tab to open the scale options in the Standard tier.
12. Slide the **DTU slider** to right to select 100 DTUs. Note this corresponds to the service objective, **S3,** shown in brackets between the DTU and Storage size meters.
13. Click **Apply** to move the database out of the pool and make it a Standard S3 database.
14. Once the deployment is complete, monitor the effect on the contosoconcert hall database and the pool it was removed from on the elastic pool and database blades.

Once the higher than normal load on the contosoconcerthall database subsides you should promptly return it to the pool to reduce its cost. If it’s unclear when that will happen you could set an alert on the database that will fire when its DTU usage drops below the per-database max on the pool. Moving a database into a pool is described in exercise 5.

## Other Performance Management Patterns

### Pre-emptive scaling

In exercise 6 where you explored how to scale an isolated database you knew which database to look for. If the management of Contoso Concert Hall had informed WTP of the impending ticket sale, the database could have been moved out of the pool pre-emptively. Otherwise, it would likely have required an alert on the pool or the database to spot what was happening. You wouldn’t want to learn about this from the other tenants in the pool complaining of degraded performance. And if the tenant can predict how long they will need additional resources you can set up an Azure Automation runbook to move the database out of the pool and then back in again on a defined schedule.

### Tenant self-service scaling

Given that scaling is an automatable task easily called via the management API, it’s quite possible to build the ability to scale tenant database into your tenant-facing application, and offer it as a feature of your SaaS service, perhaps linked to additional billing.

### Scaling a pool up and down on a schedule to match usage patterns

Where aggregate tenant usage follows predictable usage patterns, you can use Azure Automation to scale a pool up and down on a schedule. For example, scaling a pool down after 6pm and up again before 6am on weekdays.

## Resources

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-pool>

<https://docs.microsoft.com/en-us/azure/automation/automation-intro>